Mid Sem Solution

What are the pros and cons of LSTMs over RNNs (5)

1.a (5 points) Disadvantage of RNNs:

1. The vanishing gradient and gradient exploding problem.
2. Cannot be stacked into a very deep model.
3. Unable to track long-term dependencies.

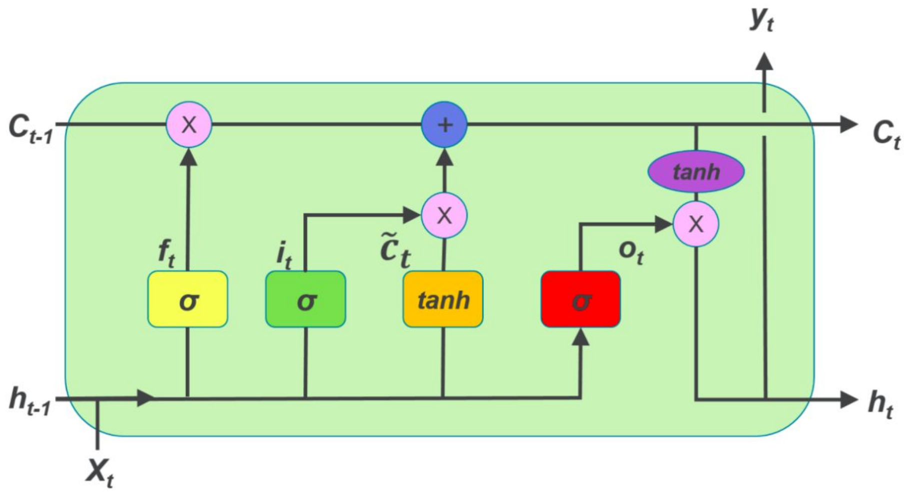
LSTM overcome these by using a memory cell as a building block.

Disadvantages of LSTM:

1. More computationally expensive due to complex memory cell.
2. Increase in the training weights due to memory cell which makes it harder to train and find an optimal solution.

1.b (10 points) List the additional components introduced in an LSTM block and describe their function by providing appropriate equations, learnable parameters and activations.

Block diagram of LSTM memory cell is shown below:

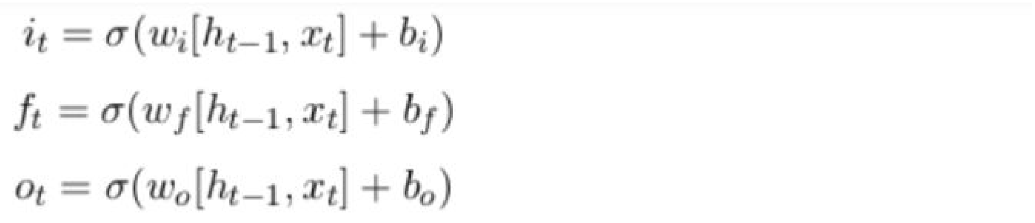


LSTM has three Gates:

1. Input Gate
2. Forget Gate
3. Output Gate.

Sigmoid functions are used as activations in gates.

The equations for different gates are:

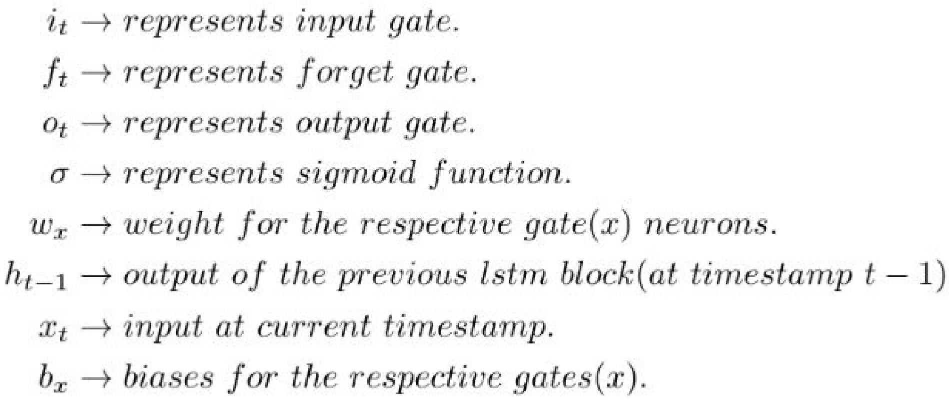


$$i_t = \sigma(w_i[h_{t-1}, x_t] + b_i)$$

$$f_t = \sigma(w_f[h_{t-1}, x_t] + b_f)$$

$$o_t = \sigma(w_o[h_{t-1}, x_t] + b_o)$$

Equation of Gates



$i_t \rightarrow represents\ input\ gate.$

$f_t \rightarrow represents\ forget\ gate.$

$o_t \rightarrow represents\ output\ gate.$

$\sigma \rightarrow represents\ sigmoid\ function.$

$w_x \rightarrow weight\ for\ the\ respective\ gate(x)\ neurons.$

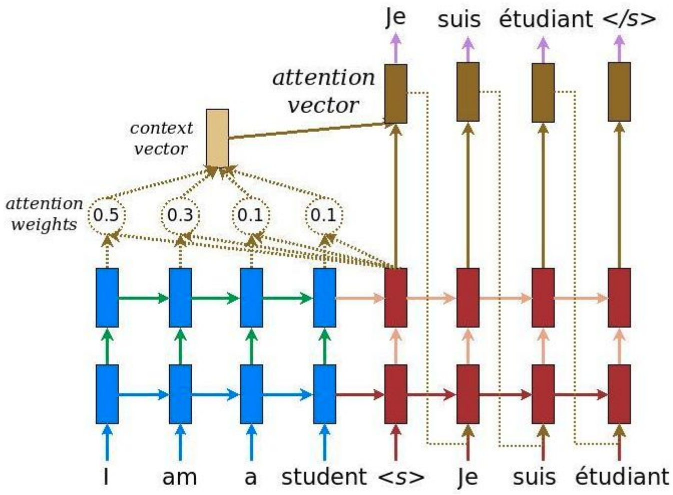
$h_{t-1} \rightarrow output\ of\ the\ previous\ lstm\ block(at\ timestamp\ t-1)$

$x_t \rightarrow input\ at\ current\ timestamp.$

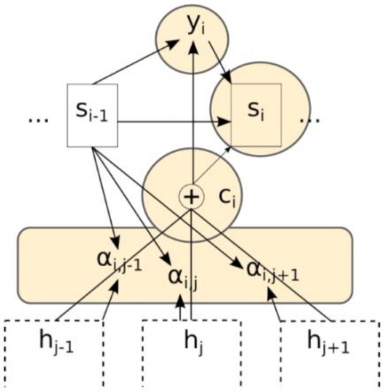
$b_x \rightarrow biases\ for\ the\ respective\ gates(x).$

1.c (2+3+5 = 10 points) What is attention? With the help of a block diagram, show how attention is used for Neural Machine Translation. Describe the attention component with necessary equations, learnable parameters and translations.

Attention allows the model to learn where to place attention on the input sequence as each word of the output sequence is decoded. Use of soft attention for neural machine translation is shown below:



Encoder and decoder are represented by blue and red respectively.



c_i := *context vector* , h_i : *Annotations* , s_i : *hidden states* , y_i : *target word*

$$e_{ij} = v^T \tanh(W s_{i-1} + V h_j)$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{L} \exp(e_{ik})}$$

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

$$s_i = f(s_{i-1}, y_{i-1}, c_i)$$

$$p(y_i|y_1, \ldots, y_{i-1}, \mathbf{x}) = g(y_{i-1}, s_i, c_i)$$

Through the attention mechanism, each time the model generates a word in a translation, it learns an association between the output words and a set of words in the source sentence. This association capture the most relevant context with respect to the predicted word.

2.a (10 points) It has been generally observed that RCNN and its variants (Fast/Faster) have better accuracy but are slower compared to YOLO or SSD. Can you argue about the reasons for it to be so.

RCNN and its variants are slower as compared to other approaches as they don't process whole image but instead works on the several regions of the individual image. For example in RCNN selective search is used to extract around 2000 regions from the image (region proposals). These region proposals are then used for objects detections. As so many region proposals are classified per image, the RCNN are slow.

In fast RCNN, instead of feeding the individual region proposals, we feed the complete image and then selective search is used on feature maps to find the region of proposals. However, the selective search is slow and hence it makes fast RCNN slow.

In faster RCNN, the selective search is avoided and a separate network is used to predict the region proposals. Still, the overhead of this separate network makes the faster RCNN slower as compared to YOLO or SSD.

On the other hand, in YOLO, for example, there are no region proposals and a single convolutional network predicts the bounding boxes and the class probabilities for these boxes. In YOLO, an image is split into multiple grids, within each of the grid we take m bounding boxes. For each of the bounding box, the network outputs a class probability and offset values for the bounding box. However, the performance of the YOLO depends on the size of the bounding boxes and YOLO face problems with detection of small objects.

2.b (4 +6 points) Why is the RoI-Pool layer necessary in Faster-RCNN? Describe the functioning of the RoI-Pool layer.

The main purpose of doing such pooling is to train the whole system from end-to-end (in a joint manner) and speed up the training and test time.

Functioning of ROI:

The RoI pooling layer uses max pooling to convert the features inside any valid region of interest into a small feature map with a fixed spatial extent of H × W (e.g., 7 × 7), where H and W are layer hyper-parameters that are independent of any particular RoI. Each RoI is defined by a four-tuple (r, c, h, w) that specifies its top-left corner (r, c) and its height and width (h, w). RoI max pooling works by dividing the h × w RoI window into an H × W grid of sub-windows of approximate size h/H × w/W and then max-pooling the values in each sub-window into the corresponding output grid cell. Pooling is applied independently to each feature map channel, as in standard max pooling.

2.c (5 points) mean Average Precision (mAP) score.What is the full-form of the evaluation metric mAP. Define it with necessary equations or formulae.

It is a metric to measure the accuracy of object detectors. It is the average of the maximum precisions at different recall values.

For a particular class, Average Precision (AP) is defined as:

$$AP = \frac{1}{N} \sum_{r} AP_r$$

where AP_r is the maximum precision at recall r. It can be simplified as:

$$AP = \frac{1}{N} \sum_{r} p_{interp}(r)$$

$p_{interp}(r)$ is the interpolated precision (p) at recall (r) and is defined as:

$$p_{interp}(r) = max\{r' > r\}p(r')$$

mAP is then defined as average of AP over all the classes.

3.a (3 points) What kind of loss function would you use for a Semantic Segmentation problem? Write the expression for the loss.

Pixel-wise cross entropy loss is The most commonly used loss function for the task of image segmentation.

$$L = -\sum_{classes} y_{true} log(y_{pred})$$

3.b (4+3 = 7 points) Comment on the need for an upsampling layer in Fully Convolutional Networks (FCNs) for semantic segmentation? Why can't we have same sized convolutional layers throughout the network and avoid the upsampling layer altogether? On the other hand, why do we need the skip connections?

In semantic segmentation the need is to have an output image of the same resolution as input. This can be achieved by stacking the filters with proper padding to maintain the same size throughout the network. However, this is computationally very expensive.

In FCN, convolutional filters and pooling layers are used to progressively downsample the input spatial resolution, developing lower-resolution feature mappings which are learned to be highly efficient at discriminating classes. Then, final features are upsampled into a full-resolution segmentation map. To restore the original size image from the downsampled features we need a learnable upsampling layer.

However, because the resolution of the input image is reduced by a large factor, producing the fine-grained segmentations from the downsampled features is difficult and this problem can be solved by using skip connections. These skip connections from earlier layers in the network (prior to a downsampling operation) provide the necessary fine detail in order to reconstruct accurate shapes for segmentation boundaries. Skip connections also help to check the vanishing gradient problem.